Today's Lecture: CFLs and Valid Computations

Reading:

Today: CFLs and Valid Computations.
Read: *Kozen* lecture 35.

November 30: Gödel's Theorem
Read: *Kozen* lecture 38 (or *Sipser* 6.2).

December 2: Something Fun: Theorem proving, etc.

I. Valid Computational Histories.

A. Let's say that Turing machine M terminates on input x. Then:

1. Let $\chi_0 = (q_0, \vdash x\Box^\omega, 0)$ be the initial configuration for the machine on input x.

2. Let $\chi_m = (q_m, \vdash y_m\Box^\omega, p_m)$ be the configuration reached after M has performed m steps:

$$\chi_0 \quad \xrightarrow[M]{m} \quad (q_m, \vdash y_m\Box^\omega, p_m)$$

3. Because m terminates on input x, there is some integer n such that

$$\chi_0 \quad \xrightarrow[M]{n} \quad (q_n, \vdash y\Box^\omega, m)$$

with $q_n \in \{t, r\}$, $y \in \Gamma^*$, and $m \in \mathbb{Z}$, where t and r are the accept and reject states for M and Γ is the tape alphabet.

4. It is easy to show (by induction on m) that for all $m \leq n$, if p_m is the position of the read/write head after m moves, then $p_m \leq m$.

5. Thus, M visits at most, the n leftmost squares of its tape when processing x.

B. Let $\Gamma' = (\Gamma \times (Q \cup \{\circ\})) \cup \{\#\}$.

1. We can now represent configurations as strings over Γ'^*. I'll write symbols in Γ' as c_q where $c \in \Gamma$ and $q \in (Q \cup \circ)$.

2. Given a configuration, $(q, y\Box^\omega, m)$ (note that the first symbol of y must be $\vdash$), we define a string y' such that each symbol in y' is the corresponding symbol in $\vdash y$ paired with $\circ$, except for the symbol in position m; we pair that one with q.

3. Formally, we can make sure that $|\vdash y| \geq m$ by padding it with $\Box$ symbols if needed. Now, define

$$\begin{array}{rcl} f(q, \epsilon, m) & = & \epsilon \\ f(q, c \cdot y, m) & = & (\text{if } m = 0 \text{ then } c_q \text{ else } c_\circ) \cdot f(y, m-1, q) \end{array}$$

4. We represent the configuration $(q, y\Box^\omega, m)$ with the string $f(q, y, m) \cdot \Box_\circ^\omega$.

C. If M accepts x in n moves, we can represent the computation that M performed as a sequence of $n+1$ strings in Γ'^{n+1}.

1. As noted above, M visits at most the first $n+1$ tape squares by the end of its n^{th} move. Thus, we only need to keep track of the first $n+1$ tape squares. The others won't affect what M does in its first n moves (even if $|x| > n$, this just means that M accepts or rejects without reading all of x).

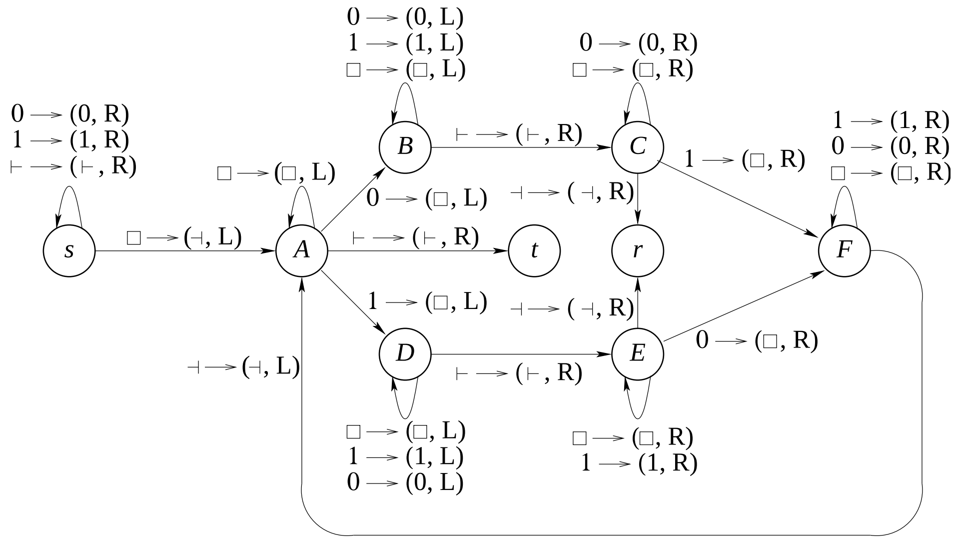

Figure 1: A Machine that accepts 0^n1^n

2. We can concatenate the strings for the $n+1$ configurations together, using the $\#$ symbol as a separator, to get a string in Γ'^{n+1}.

3. An example:

a. Let M be a machine with input alphabet $\{0,1\}$ that accepts strings that have the same number of 0's and 1's. Figure 1 shows such a machine.

b. Here's a brief description of how the machine works.

i. In state s it scans to the right until it encounters the first $\Box$. It replaces this with an right endmarker, $\dashv$, and moves to state A.

ii. The machine now repeatedly makes right-to-left and left-to-right scans. On each scan, it erases one 0 symbol and one 1 symbol. If it reaches a point where all of the 0's and 1's have been erased, it accepts. Otherwise, if there are 0's left over after all of the 1's are erased (or vice versa), it rejects. I explain the details in the following.

iii. If the first non-blank character that it encounters on a right-to-left scan is a 0, the machine erases the 0, enters state B, and completes moving to the left until it reaches the left endmarker. It then enters state C. In state C, the machine moves to the right looking for a 1. If it finds one, then it erases it and enters state F. Otherwise, it rejects the string because there are more 0's than 1's.

iv. If the first non-blank character that it encounters on a right-to-left scan is a 1, the machine erases the 1 and enters state D. It completes the scan to the left, turns around, and looks for a 1. If it finds one, it erases it and enters state F. Otherwise, it rejects the string because there are more 1's than 0's.

v. If the first non-blank character that it encounters is the left endmarker, then it has successfully erased all of the 0's and 1's and accepts.

vi. When the machine enters state F, it completes the left-to-right scan, and returns to state A for the next iteration.

c. Figure 2 shows the sequence of configurations that the machine goes through to reject the input 110. It also shows how this can be represented by a long string over the alphabet

$$(\{0,1,\Box,\vdash,\dashv\} \times \{s,t,r,A,B,C,D,E,F,\circ\}) \cup \{\#\}$$

In particular, the machine from Figure 1 takes 21 steps to reject

step		configuration		$string$
0.		$(s, \vdash 110\square^{\omega},\ 0)$		$\# \vdash_s 1_\circ 1_\circ 0_\circ \square_\circ^{17}$
1.	$\xrightarrow[M]{1}$	$(s, \vdash 110\square^{\omega},\ 1)$	$\cdot$	$\# \vdash_\circ 1_s 1_\circ 0_\circ \square_\circ^{17}$
2.	$\xrightarrow[M]{1}$	$(s, \vdash 110\square^{\omega},\ 2)$	$\cdot$	$\# \vdash_\circ 1_\circ 1_s 0_\circ \square_\circ^{17}$
3.	$\xrightarrow[M]{1}$	$(s, \vdash 110\square^{\omega},\ 3)$	$\cdot$	$\# \vdash_\circ 1_\circ 1_\circ 0_s \square_\circ^{17}$
4.	$\xrightarrow[M]{1}$	$(s, \vdash 110\square^{\omega},\ 4)$	$\cdot$	$\# \vdash_\circ 1_\circ 1_\circ 0_\circ \square_s \square_\circ^{16}$
5.	$\xrightarrow[M]{1}$	$(A, \vdash 110 \dashv \square^{\omega},\ 3)$	$\cdot$	$\# \vdash_\circ 1_\circ 1_\circ 0_A \dashv_\circ \square_\circ^{16}$
6.	$\xrightarrow[M]{1}$	$(B, \vdash 11\square \dashv \square^{\omega},\ 2)$	$\cdot$	$\# \vdash_\circ 1_\circ 1_B \square_\circ \dashv_\circ \square_\circ^{16}$
7.	$\xrightarrow[M]{1}$	$(B, \vdash 11\square \dashv \square^{\omega},\ 1)$	$\cdot$	$\# \vdash_\circ 1_B 1_\circ \square_\circ \dashv_\circ \square_\circ^{16}$
8.	$\xrightarrow[M]{1}$	$(B, \vdash 11\square \dashv \square^{\omega},\ 0)$	$\cdot$	$\# \vdash_B 1_\circ 1_\circ \square_\circ \dashv_\circ \square_\circ^{16}$
9.	$\xrightarrow[M]{1}$	$(C, \vdash 11\square \dashv \square^{\omega},\ 1)$	$\cdot$	$\# \vdash_\circ 1_C 1_\circ \square_\circ \dashv_\circ \square_\circ^{16}$
10.	$\xrightarrow[M]{1}$	$(F, \vdash \square 1\square \dashv \square^{\omega},\ 2)$	$\cdot$	$\# \vdash_\circ \square_\circ 1_F \square_\circ \dashv_\circ \square_\circ^{16}$
11.	$\xrightarrow[M]{1}$	$(F, \vdash \square 1\square \dashv \square^{\omega},\ 3)$	$\cdot$	$\# \vdash_\circ \square_\circ 1_\circ \square_F \dashv_\circ \square_\circ^{16}$
12.	$\xrightarrow[M]{1}$	$(F, \vdash \square 1\square \dashv \square^{\omega},\ 4)$	$\cdot$	$\# \vdash_\circ \square_\circ 1_\circ \square_\circ \dashv_F \square_\circ^{16}$
13.	$\xrightarrow[M]{1}$	$(A, \vdash \square 1\square \dashv \square^{\omega},\ 3)$	$\cdot$	$\# \vdash_\circ \square_\circ 1_\circ \square_A \dashv_\circ \square_\circ^{16}$
14.	$\xrightarrow[M]{1}$	$(A, \vdash \square 1\square \dashv \square^{\omega},\ 2)$	$\cdot$	$\# \vdash_\circ \square_\circ 1_A \square_\circ \dashv_\circ \square_\circ^{16}$
15.	$\xrightarrow[M]{1}$	$(D, \vdash \square\square\square \dashv \square^{\omega},\ 1)$	$\cdot$	$\# \vdash_\circ \square_D \square_\circ \square_\circ \dashv_\circ \square_\circ^{16}$
16.	$\xrightarrow[M]{1}$	$(D, \vdash \square\square\square \dashv \square^{\omega},\ 0)$	$\cdot$	$\# \vdash_D \square_\circ \square_\circ \square_\circ \dashv_\circ \square_\circ^{16}$
17.	$\xrightarrow[M]{1}$	$(E, \vdash \square\square\square \dashv \square^{\omega},\ 1)$	$\cdot$	$\# \vdash_\circ \square_E \square_\circ \square_\circ \dashv_\circ \square_\circ^{16}$
18.	$\xrightarrow[M]{1}$	$(E, \vdash \square\square\square \dashv \square^{\omega},\ 2)$	$\cdot$	$\# \vdash_\circ \square_\circ \square_E \square_\circ \dashv_\circ \square_\circ^{16}$
19.	$\xrightarrow[M]{1}$	$(E, \vdash \square\square\square \dashv \square^{\omega},\ 3)$	$\cdot$	$\# \vdash_\circ \square_\circ \square_\circ \square_E \dashv_\circ \square_\circ^{16}$
20.	$\xrightarrow[M]{1}$	$(E, \vdash \square\square\square \dashv \square^{\omega},\ 4)$	$\cdot$	$\# \vdash_\circ \square_\circ \square_\circ \square_\circ \dashv_E \square_\circ^{16}$
21.	$\xrightarrow[M]{1}$	$(r, \vdash \square\square\square \dashv \square^{\omega},\ 5)$	$\cdot$	$\# \vdash_\circ \square_\circ \square_\circ \square_\circ \dashv_\circ \square_r \square_\circ^{15} \#$

Figure 2: Configurations for the machine from Figure 1 when rejecting input 110

.

II. Undecidable problems for CFLs.

A. Let M be a Turing machine, and let x be a string. Does M halt on x?

1. We can use the computational histories defined above to examine this question.

a. If M halts with input x, then there is a some integer n, such that there is a string of length $(n+1)(n+2)+1$ symbols that describes the computation.

b. The $n+1$ is for the configurations χ_0 through χ_n.

c. The machine visits at most $n+1$ squares of the tape, and configurations are separated by the $\#$ symbol, thus we can write each configuration with exactly $n+2$ symbols.

d. The final $+1$ is because Kozen surrounded each configuration with $\#$ symbols, and I'll follow his example.

2. Let α be a string in Γ'^*. What properties must α have if it describes a valid, halting computation?

a. It must be of the form $\#\alpha_0\#\alpha_1\#\ldots\#\alpha_n\#$.

b. Each α_i must be of the form: ${\beta_\circ}^*\beta_q{\beta_\circ}^*$, where $\beta_\circ$ matches any symbol in $\Gamma \times \{\circ\}$, and β_q matches be any symbol in $\Gamma \times Q$.

c. α_0 represents the initial configuration with input x. In other words, $\alpha_0 = \vdash_s x_\circ \square^*$, where $x_\circ$ is the string in Γ'^* corresponding to x with every symbol in x paired with $\circ$.

d. α_n represents a configuration in a final state of M: ${\beta_\circ}^*\beta_{tr}{\beta_\circ}^*$, where β_{tr} matches any symbol in $\Gamma \times \{t, r\}$.

e. The string α_{i+1} is the valid successor of α_i according to the relation $\xrightarrow[M]{1}$.

We note that the first four properties correspond to a regular language corresponding to the regular expression:

$$\# \vdash_s x_\circ \square^* (\#\beta_\circ^*\beta_q{\beta_\circ}^*)\#\beta_\circ^*\beta_{tr}{\beta_\circ}^*\#$$

We'll show that the fifth property is the complement of a context-free language.

B. Revisiting an old friend, who's context-free

1. Recall $A = \{x| \; \exists w. \; x = ww\}$ is not context-free, but $\sim\!A$ is a CFL. We can recognize $\sim\!A$ with the following PDA:

a. If $y \in A$, then either $|y|$ is odd, or we can find symbols c and d, and strings u, v, w, and x, such that $y = ucvwdx$, $d \neq c$, $|u| = |w|$, and $|v| = |x|$. In other words, if y isn't the repetition of some string, then the first and second half of y must be different. This means that they differ in at least one position. The symbols c and d are these symbols that differ. The strings u, v, w, and x just keep track of how far we are into each string to make sure that c and d came from corresponding positions.

b. A PDA can recognize language $\sim\!A$ by

i. Pushing a marker on for each symbol in u.

ii. Remembering the symbol c in its finite state.

iii. Popping markers off until the top-of-stack marker, $\perp$ is revealed, and then pushing on markers until it reaches symbol d. Note that $|v| + |w| = |u| + |x|$. Thus, there are $|w|$ markers on the stack at this point.

iv. Verify that $d \neq c$.

v. Pop markers off the stack until the top-of-stack marker is uncovered again.

vi. If it has consumed the entire input string, it has shown that the y is not of the form ww.

vii. Note that the machine uses non-determinism to "guess" where c and d are, but the counting that it does by pushing and popping markers proves that it didn't cheat.

2. Let $B = \{x| \; \exists w \in (\Sigma - \{\#\})^*. \; x = \#w\#w\#\}$. We can show that B is not context free, but that $\sim\!B$ is context-free by pretty much the same construction as before.

a. A string, y, is in $\sim\!B$ iff at least one of the following four conditions applies

i. y does not contain three $\#$ symbols. This is a regular language, therefore it is context-free.

ii. y has other symbols before the first $\#$ or after the last $\#$. Again, this is regular and therefore context-free.

iii. The two strings between the $\#$ symbols are of different lengths. This is context-free (a PDA can count the symbols using markers on its stack and accept if the counts don't match).

iv. The two strings between the $\#$ symbols differ. We can make a PDA that ignores the $\#$ symbols, and this becomes the problem of showing that y is not of the form ww. We just showed that this is context-free.

Thus, $\sim B$ is the union of four context-free languages. Context free-languages are closed under union. Therefore, $\sim B$ is context free.

3. Let $C = \{x | \ \exists w \in (\Sigma - \{\#\})^*. \ x = \#(w\#)^*\}$ Again, C is not a CFL, but $\sim C$ is. Given an input string $\#x_0\#x_1\# \ldots \#x_n\#$, a PDA can non-deterministically guess a consecutive pair of x's that don't match, and verify that guess using the procedure described above.

4. Note that, rather than checking that $d \neq c$, we can check that $d \neq f(c)$ for any function that we like. We can also check to see if there is some sequence of three symbols, $c_1c_2c_3$ such that the symbols in the corresponding position in the second string, $d_1d_2d_3$ don't match $f(c_1c_2c_3)$. We'll define f to track match what a Turing machine does. Let $\alpha_i = \beta_i c_{1,\circ} c_{2,q} c_{3,\circ} \gamma_i$.
   - **a.** If $(q, c_2) \rightarrow (q', e, L)$, then let $d_1 = c_{1,q'}$, $d_2 = e_\circ$, and $d_3 = c_{3,\circ}$.
   - **b.** Otherwise $(q, c_2) \rightarrow (q', e, R)$, and we let $d_1 = c_{1,\circ}$, $d_2 = e_\circ$, and $d_3 = c_{3,q'}$.

   To show that two consecutive configurations are *not* valid successors in a computation of m, we find α_i and α_{i+1} such that $\alpha_i = \beta_i c_{1,\circ} c_{2,q} c_{3,\circ} \gamma_i$. and $\alpha_{i+1} \neq \beta_i d_1 d_2 d_3 \gamma_i$. As described above, we can check this with a PDA. Aside from the usual stuff of checking the lengths of α_i and α_{i+1}, and making sure that α_i only has one symbol marked with a state (rather than with $\circ$), the machine looks for
   - **c.** Symbols in corresponding positions of α_i and α_{i+1} that don't match, and that are not within distance one of the symbol marked with the machine state in α_i.
   - **d.** It looks for symbols marked with the machine state in α_i and its immediate left and right neighbours and determines that the symbols in the corresponding position in α_{i+1} don't correspond to the appropriate move of M.

C. An undecidable problem for CFLs

1. We've shown that the first four conditions for a valid computation history form a regular language. Therefore, their complement is a regular language and is context-free.

2. We've shown that the complement of the final condition is context-free. Basically, a PDA can guess which pair of configurations is "wrong." If some symbol away from the tape head has been altered, then it detects that the same way a PDA can show that a string is not of the form ww. On the other hand, if the string doesn't correspond to the right move, the PDA can remember the symbol under the head of α_i, its left and right neighbours, and the TM state in the PDA's state. It then checks the corresponding three symbols for α_{i+1} and shows that they don't match up.

3. Let G be the CFG that corresponds to invalid computations.
   - **a.** If $L(G) = \Gamma'^*$, then there is no valid computation that leads to a final state. We conclude that M does not terminate on input x.
   - **b.** On the other hand, if $L(G) \neq \Gamma'^*$, then let $z \in \sim L(G)$. The string z describes a terminating computation on x. Thus, M halts on input x.
   - **c.** The question of whether or not M halts on input x can be reduced to the question of whether or not the language of a context-free grammar is Γ'^*.
   - **d.** Thus, the question of whether or not a context-free grammar generates all strings is undecidable.